

VizRoom User's Manual

Mark Green
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada

Introduction

The document is an introduction to the VizRoom that concentrates on how this facility is used. It includes a description of the VizRoom hardware, how to run programs in the VizRoom, and an overview of the operation of VizRoom programs. There is also a short trouble shooting section covering some of the basic problems that can occur when using the VizRoom. This document doesn't cover any of the programming details. This information can be found in other VizRoom documents.

Facility Overview

The VizRoom's major hardware components are described in this section. The purpose of these descriptions is to give the reader a basic understanding of the hardware devices and how they are used.

The VizRoom is a 9.5' x 9.5' room that consists of three walls constructed from a projection screen. Three projectors are used to present the images generated by an SGI Oynx2 computer. While in the VizRoom the position and orientation of the user is tracked, so the projected images correspond to the user's current point of view. Some applications use stereo graphics, and for these application stereo glasses must be worn to see the effect. The main aim of all this technology is to immerse the user in a 3D space where he or she can directly interact with a computer generated visualization.

The following sections cover the main components of the VizRoom, and provide all the information that most users need to know about interacting with them.

Oynx2 Computer

An SGI Oynx2 computer, called cadomin, generates all the graphics in the VizRoom. This computer has 4 processors and 512 Mbytes of main memory. It currently has 27 Gbytes of disk space as well. The disks are organized into several file systems. When you get an account for the VizRoom you will be given a home directory on one of the standard file systems. These file systems are backed up on a regular basis, so this is a good place to store your programs and small data files. There is also a larger file system that isn't backed up. This file system is currently called cadomin6, and you should place all your large data files on this file system.

There is a very high-speed connection between cadomin and aurora. This is a HIPPI connection, and at approximately 800 Mbits/sec. This connection can be used to transfer files from aurora, or directly connect to computations running on that machine.

Cadomin currently has two InfiniteReality graphics pipes. Each of these pipes operates as a separate display device, and each one is capable of displaying over 10 million polygons per second. Each pipe can produce two video signals. Pipe 0 is used for the front wall of the VizRoom and the CRT monitor, while pipe 1 is used for the two side walls. Cadomin can produce a number of different video formats, ranging from standard NTSC (in both analogue and digital form) through HDTV. The output from cadomin can also be sent to any workstation or PC that supports the GLX protocol. Most recent SGI workstations and PCs that use Exceed support this protocol.

Cadomin runs the most recent version of Irix, which is SGI's version of Unix. You can telnet to cadomin from anywhere on campus, and use this connection for program development. Please don't run large jobs on cadomin, since they will have a major impact on whoever is using the VizRoom. Also note that cadomin has the same basic machine architecture as aurora, so its fairly easy to move programs between these two computers.

Projectors

Three Electrohome Marquee 8500 projectors are used to produce the images on the VizRoom walls. These are CRT projectors that are capable of projecting high-resolution stereo graphics. They have a special green tube that is necessary for stereo graphics. These projectors are quite delicate and should never be touched. They must be positioned accurately to exactly cover the screens and meet in the corners. In addition there are special calibration procedures required to focus and converge them. It can take several days to re-calibrate the projectors if they are moved or the settings are modified. There is no reason to touch the projectors themselves, so please don't attempt to adjust them yourself.

The projectors are normally left in a standby state where the CRTs are turned off, but the rest of the electronics stays running. This standby state extends the life of the CRTs. If the projectors are turned off completely, it can take up to 45 minutes for them to warm up and the images to stabilize. This is why they are left in the standby state and never turned off. There is a program that can be used to switch the projectors in and out of standby mode and this program should be used instead of interacting directly with the projectors themselves.

Screens

The three walls of the VizRoom are constructed from a single piece of back projection screen. This screen has a plastic base so it can be wrapped around the corners of the room. The screen is attached to an aluminum frame, and Weed Whacker cord is used to form the corners. Don't touch the screen. The screen can be damaged if you press on it

or hit it with a sharp object. It can also be stretched out of shape if you push on it. The screen should be kept as flat as possible to produce the best possible images.

Trackers and Input Devices

In the standard VizRoom configuration two electromagnetic trackers are used to determine the user's position in space and interact with programs. These trackers are based on a source that generates a time varying electromagnetic field. The source is the large gray cube mounted above the center of the VizRoom. The two sensors measure the electromagnetic field and use these measurements to determine their position and orientation in space.

The tracker without buttons is used for head position and orientation. There is a strip of Velcro on the bottom of this tracker so it can be attached to a baseball hat. The user wears the hat, and the images projected on the walls follow his or her motions. The other tracker is used for interacting with the application. The buttons can be used for entering commands, and their interpretation depends upon the program that is currently running.

We are working on a number of other input devices that will make the VizRoom easier to use. Some of these devices are based on image processing techniques and speech recognition.

Quick Start

This section covers the basic procedure for running applications in the VizRoom. This should provide all the information you need to get started with most VizRoom applications. Some applications require extra steps, which are covered in their documentation. If you are using devices other than the standard trackers, there may also be some special procedures that may need to be followed.

The steps are first presented in list form for easy reference, followed by a description of the purpose of each step.

1. Log on to Oynx2 using the main console. The main console has a large HDTV monitor, so it's hard to miss.
2. Check that the projectors are on and not in the standby state. The easiest way to do this is through the *projectors* program. Type *projectors on* at the command prompt to ensure that the projectors are ready to use. Note that most of the programs mentioned in this document can be found in the /usr/MR/bin directory. When your account was set up this directory was placed on your search path.
3. You are now ready to run your program. If your program is a multiprocessor program, just run the program in the normal way (type the name of the program and its parameters). If the program is a single processor program you need to use the *runon* command to execute it. This forces the program to run on a specific processor. There are four processors on the Oynx2, numbered 0 through 3. You specify the

processor as the first parameter to the *runon* command, followed by the name of the program and its parameters.

4. Once you are finished using the VizRoom put the projectors back in standby mode (unless someone else is waiting to use it). Again the *projectors* program to do this by typing *projectors off* at the command line.

You must first log into the Oynx2 console before you can run a program in the VizRoom. If there is no one on the Oynx2 console you won't be able to run your program. VizRoom programs use the X server that is running on the console and if no one is on the console this server won't be running.

As noted in the previous section, the projectors are normally in the standby mode to prolong the life of the CRTs. The projectors program can be used to control the projectors, and this is the easiest way to switch them into and out of standby mode. The projectors program also checks that the projectors are powered on and responding correctly to the Oynx2. It's not a good idea to try to adjust the projectors yourself.

There are two types of programs that can be run in the VizRoom. A multiprocessor program takes advantage of the multiple processors and graphics pipelines on the Oynx2 to generate the images for several walls at the same time. This is the most efficient way to run the VizRoom, but it requires some extra programming. If you are using a multiprocessor program you only need to enter the name of the program and its parameters on the command line.

Some VizRoom programs use a single processor to generate all their images. These programs typically don't use a lot of graphics, or producing a multiprocessor version of them requires a considerable amount of programming. There is a bug in the Oynx2 that causes some problems for these programs. If they are run in the same way as multiprocessor programs there is a good chance that they will crash the system. This will cause the entire machine to reboot, which takes 5 or 10 minutes, and everyone using it will lose their work. This is obviously not a very good situation, but there is a fix. If the *runon* command is used the program will be forced to run on a specific processor for the entire length of the run. This seems to solve the problem. The first parameter to the *runon* program is the processor to run the program on, and the rest of the line is the name of the program and its parameters.

Booking the VizRoom

The VizRoom is available 24 hour a day, seven days a week. It is located in GSB 602 and a key can be obtained for accessing this room after hours. Keys will only be issued to researchers who have attended our "Introduction to the VizRoom" course, or have received similar training. There is normally someone in the room from 10am until 5pm, Monday through Friday to assist you with using the VizRoom.

There is a booking procedure for the VizRoom, which can be used to reserve time for your use. The booking procedure is web based and can be accessed from our Web page:

<http://www.cs.ualberta.ca/~graphics/vizroom>. Bookings have priority over other uses, but if the VizRoom isn't booked feel free to drop in and use it on a first come first served basis. On rare occasions a booking may be cancelled due to hardware or software problems, or the need to demo the facility to funders. In this case, the effected parties will be notified as soon as possible.

Theory of Operation

This section presents an overview of what happens when a program runs in the VizRoom. It doesn't cover all of the technical details, but provides enough information to determine why your program may not be operating the way you expect it to.

The software packages that are used in the VizRoom are general virtual reality packages that have not been specially designed to drive this type of facility. The programs that are developed with these packages can run on a variety of equipment configurations, so one of the first things that these programs must do is determine the current equipment configuration. There are a number of text files that are used for this purpose. These files are normally found in the /usr/MR/etc directory, but there could be local copies in the directory that you are running from. If there are local versions, they are typically similar to the standard ones with a few modifications to support the program that you are using.

The most important configuration file is the mapping file, it's the one that's customized the most often. If there is a local mapping file, it will have the name of the application plus the suffix .map. For example, if the application is myapp then the mapping file will be called myapp.map. If there isn't an application specific mapping file in the current directory, /usr/MR/etc/MRmap is the one that is usually used. Users can have their own local mapping files in their home directory and in a few other places, but this is rarely done.

An application uses the mapping file to determine the real devices it's to use. The application is written in terms of logical devices, and at run-time these logical devices are mapped into real devices using the mapping file. In the case of input devices, the mapping file contains the name of the server for the device and any parameters that it requires. In the case of output devices, the mapping file contains a description of the device. In the case of the VizRoom, there are at least three output devices in the mapping file, corresponding to the three walls of the VizRoom. These entries describe how to connect to the display device, its resolution and whether stereo is used. The mapping files can also specify that graphical output is to be sent to another computer.

There is a separate document that describes the structure of mapping files, and this document should be referenced if you are interested in constructing your own or modifying existing ones.

In the case of input devices, the mapping file contains the name of the server that controls the device. We use a client-server model for most of our input devices so they can be accessed from several different workstations. Once the name of the server has been determined, the application must determine how to connect to that server. This

information is contained in the `servertab` file. This file is normally found in the `/usr/MR/etc` directory, but there could be a local copy in the current directory. This file has one line for each input device. This entry contains the address of the machine the runs the server and the port used to connect to the server. It also contains several other pieces of information that you don't need to worry about. Occasionally devices will be moved from one computer to another. When this occurs the copy of `servertab` in `/usr/MR/etc` is modified. If you have a local copy of `servertab` and a device is moved, your application won't be able to find the device until your local copy is updated.

The last file that plays a role in the configuration process is the workspace file. Again this file is normally located in the `/usr/MR/etc` directory, but you can have a local copy. This file is used to convert from device specific coordinates into a common coordinate system. Some input devices define their own coordinate system and report all their measurements in this system. For example, the electromagnetic trackers used in the VizRoom have their own coordinate system that is defined by the location and orientation of the source. If several input devices are used in the same application there must be a common coordinate system or they can't be used together. Similarly, if a device is moved, the mapping from its coordinate system to the real world will change. All of these details are handled by the workspace file, which has an entry for each input device. This entry contains the location of the device within the room (the common coordinate system) and the directions of its coordinate axis.

Once the device configuration has been determined, the application connects to all of the devices and starts the main application loop. Some applications reconfigure the graphics pipes to handle stereo and other settings. This could result in the console going into an unusable state. When these programs exit, they should return the console to a usable state. If this doesn't occur, type `/usr/gfx/setmon normal` at any command prompt. Once the displays have been initialized the application enters a loop where it reads the input devices, performs application computations and updates the displays. For good interactive response this loop should be executed at least 10 times per second.

There are basically two techniques that are used to produce the graphics. In the simplest approach a single process is used and the walls are updated sequentially. That is the application produces the graphics for one wall at a time. If the data to be displayed is fairly simple this produces good results with a minimal amount of programming. But this isn't the most efficient way to use the Oynx2. The current configuration has four processors and two graphics pipes, so two walls can be updated at the same time. This can require considerably more programming. In this approach typically three processors are used. One processor is responsible for reading the input devices and performing any application computations, while the other two processors are used for graphics. There must be some synchronization between these processors, since the application data shouldn't be displayed while it is being updated.

Trouble Shooting

There are several things that could prevent a VizRoom program from running. This section covers some of the basic problems and how you can fix them. There is no

particular order to the items. Note if your problem isn't on the list, or the suggestions don't fix it, get in touch with an expert. If there is no one else in the laboratory send email to either lloyd@cs.ualberta.ca or mark@cs.ualberta.ca.

1. There are images on the console, but I don't see anything on the VizRoom walls?
The most likely cause of this problem is that the projectors are either turned off or still in standby mode. Enter *projectors on* at the console to solve this problem. It takes around 1 minute for the projectors to get up to full brightness level. Another possibility is that the mapping file is using a format that the projectors can't handle or there is no entry for the projectors in the mapping file. This will only be the case of there is an application specific mapping file.
2. The images on the screen are moving about wildly and seem to be distorted. The usual cause of this is the head tracker. The electromagnetic trackers have a limited range, and if the sensor is more than about 5 feet away from the source it will start producing very noisy values. The best way to solve this problem is to move the tracker into the VizRoom. If you won't be moving around very much you can hang the tracker from the source.
3. Everything was working okay and all of a sudden all the graphics disappeared. Most likely the screen save turned on. Go to the console mouse and move it. Everything should return to normal fairly soon (give the projectors a few second to get back to full strength).
4. I can't see stereo with the shutter glasses. First make sure that the application you are running actually produces stereo graphics. Some applications don't use stereo in order to increase update rate. If you are running a stereo application, take off the glasses, close them and open them again. When you open them point them at one of the IR emitters. Be careful not to put your fingers on the lens.
5. The images don't meet at the corners. Sometime the projectors don't synch correctly to the input signal. Try using the *projectors* program to turn the projectors off and then on again. If this doesn't solve the problem seek assistance, don't try to fix the projectors yourself.
6. The program won't start running. There are several reasons why this might happen. One of the main reasons is that that application can't access one of the input devices that it needs. First, check that no one is using the PC next to the screens. This PC runs the tracker software. Make sure that a user is logged in. If not, log in as guest (no password). This will automatically start up the tracker software. If there is a local copy of the *servertab* file in the application's directory, check that it is referencing valid servers. Compare this file to the standard one in */usr/MR/etc*. Try moving the local copy of *servertab* to another directory or give it another name. If the program work now, then the problem is with the local copy of *servertab*.